```
#include<bits/stdc++.h>
using namespace std;
int redundant(int n);
void disp(vector<int>);
int isPowerOfTwo(int);
void hamming(vector<int>, int, int);
void hammingCheck(vector<int>);

int main() {
    int ch;
    char choice;
    do{
        cout<<"1. Create hamming code";
        cout<<"\n2. Check hamming code";
        cout<<"\nEnter your choice: ";
        cin>>ch;

        switch(ch) {
            case 1:
            {
                cout<<"Enter number of bits is i/p: ";
                int n; cin>>n;
                vector<int> data(n);


                cout<<"Enter the data: ";
                for(int i=0; i<n; i++) {
                    cin>>data[i];
                }

                int redBits = redundant(n);
                hamming(data, n, redBits);
                break;

            }

            case 2:
            {
                cout<<"Enter number of bits is i/p: ";
                int n; cin>>n;
                vector<int> dataCheck(n);

                cout<<"Enter the data: ";
                for(int i=0; i<n; i++) {
                    cin>>dataCheck[i];
                }

                hammingCheck(dataCheck);
                break;
            }


            default : cout<<"Invalid Choice";
        }
```

```
        cout<<"Do you want to continue? (y/n)";
        cin>>choice;
    }while(choice=='y' || choice=='Y');
    return 0;
}

int redundant(int n){
    //O(n)
    for (int i=0; i<n; i++) {
        if(pow(2, i) >= n+i+1)
            return i;
    }
}

void hamming(vector<int> data, int n, int redBits) {
    vector<int> ans;
    list<int> red;
    int j=0;

    for(int i=(n+redBits); i>=1; i--) {
        if(isPowerOfTwo(i)) {
            red.push_front(i);
            ans.push_back(0);
        }
        else{
            ans.push_back(data[j]);
            j++;
        }
    }
    cout<<"Before chechking: ";
    disp(ans);

    int parity;
    int bitStatus;
    int cnt=0;

    for(int index: red) {
        parity=0;//initial value
        for(int i=1; i<=(n+redBits); i++) {

            bitStatus = (i >> cnt) & 1;
            if(bitStatus == 1) {
                //calculating parity(even) by usig XOR
                parity = parity ^ ans[ans.size()-i];
            }
        }
        ans[ans.size()-index] = parity;
        cnt++;
    }

    cout<<"After inserting parity values: ";
    disp(ans);
}
```

```
void disp(vector<int> ans) {

    for(int val: ans)
        cout<<val<<" ";
    cout<<endl;
}

int isPowerOfTwo(int num) {
    if(num == 1)
        return 1;
    //O(sqrt(num))
    while (num>1){
        if(num%2!=0)
            return 0;
        num=num/2;
    }
    return 1;
}

void hammingCheck(vector<int> data) {
    list<int> redBits;
    for(int i=1; i<=data.size(); i++) {
        if(isPowerOfTwo(i)) {
            redBits.push_front(i);
        }
    }

    vector<int> parity(redBits.size());
    int bitStatus, cnt =  0;
    bool flag=false;
    for(int index: redBits) {
        for(int i=1; i<=data.size(); i++) {
            bitStatus=(i >> cnt) & 1;
            if(bitStatus==1) {
                parity[cnt] = parity[cnt] ^ data[data.size()-i];
            }
        }
        if(parity[cnt] == 1)
            flag=true;
        cnt++;
    }
    if(!flag){
        cout<<"There is no error"<<endl;
    }
    else {
       int index=0;
       for(int i=parity.size()-1; i>=0; i--) {
           index = index + (parity[i]<<i);
           cout<<parity[i]<<" ";
       }
       cout<<endl;
       cout<<"Error is at index: "<<index<<endl;
    }
```

```
}

/*  -- sample output --

            1. Create hamming code
            2. Check hamming code
            Enter your choice: 1
            Enter number of bits is i/p: 9
            Enter the data: 1 0 1 1 0 0 1 1 1
            before chechking: 1 0 1 1 0 0 0 1 1 0 1 0 0
            After inserting parity values: 1 0 1 1 0 1 0 1 1 1 1 0 0
            Do you want to continue? (y/n)y
            1. Create hamming code
            2. Check hamming code
            Enter your choice: 2
            Enter number of bits is i/p: 13
            Enter the data: 1 0 1 1 0 1 0 1 1 1 1 0 0
            There is no error
            Do you want to continue? (y/n)n
*/
```